Abhängigkeiten zwischen Objekten in ingenieurwissenschaftlichen Anwendungen

Jochen Hanff
jochen@ifb.bv.tu-berlin.de
Institut für Bauingenieurwesen
TU Berlin

1 Problemstellung

Objektorientierte Anwendungen speichern Daten in strukturierten Mengen, deren Elemente Objekte sind. Ein Objekt besteht aus Methoden und Attributen. Die Daten eines Objektes werden in seinen Attributen gespeichert. Der Wert eines Attributes ergibt sich aus einer Eingabe des Benutzers oder wird mit einer algebraischen Regel aus den Werten anderer Attribute berechnet. Dadurch ergeben sich Abhängigkeiten zwischen den Attributen.

Objekte des Ingenieurwesens zeichnen sich durch eine hohe Komplexität und vielfältige Beziehungen zu anderen Objekten aus. Der Rechenaufwand für die Aktualisierung von Objekten und ihren Attributen ist in der Regel sehr hoch. Die Aktualisierung einer Objektbasis wird deshalb erst nach Änderung mehrerer Attribute zu einem vom Bearbeiter festgelegten Zeitpunkt durchgeführt. Die verzögerte Aktualisierung der Objektbasis einer Anwendung ist typisch für ingenieurwissenschaftliche Anwendungen.

Die Beziehungen zwischen den Objekten sind statisch und dynamisch. Statische Beziehungen sind zur Entwicklungszeit einer Anwendung bekannt und können deshalb zur Entwicklungszeit im Programmcode festgelegt werden. Dynamische Beziehungen sind erst zur Ausführungszeit des Programms bekannt und werden vom Benutzer zur Laufzeit der Anwendung erzeugt und entfernt.

Aufgrund der großen Anzahl von Objekten einer Anwendung aus dem Ingenieurwesen und der Vielzahl der Beziehungen zwischen den Objekten ist eine systematische Vorgehenweise auf Grundlage der System- und Graphentheorie bei der Aktualisierung einer Objektbasis zweckmäßig.

2 Systeme

2.1 Elemente

Menge und Element Objekte unserer Anschauung oder unseres Denkens, die trennbar und eindeutig identifizierbar sind, heißen Elemente. Die Zusammenfassung von Elementen zu einem Ganzen heißt Menge.

Basiselement Ein Element, das nicht aus anderen Elementen zusammengesetzt ist, heißt Basiselement. Ein Basiselement wird mit einem eindeutigen Symbol bezeichnet. Dieses Symbol heißt Identifikator des Basiselements.

Basismenge Eine Zusammenfassung von Basiselementen heißt Basismenge. Eine Basismenge wird durch Aufzählung der Identifikatoren der Basiselemente beschrieben.

Basiseigenschaft und Basiswert Eine Abbildung eines Elementes auf eine Basismenge M_B heißt eine Basiseigenschaft des Elementes. Das Bild eines Elementes unter einer Basiseigenschaft f heißt der Basiswert des Elementes für f. Der Basiswert wird mit dem Identifikator des Basiselementes bezeichnet.

Zusammengesetzte Eigenschaft Gegeben sei eine Menge M von Elementen. Für jedes Element x in M seien die Basiseigenschaften $f_1, ..., f_n$ definiert. Das Tupel $(f_1, ..., f_n)$ heißt zusammengesetzte Eigenschaft f des Elementes x.

$$\begin{array}{lll} f & := & (f_1, ..., f_n) \\ f & & \text{Zusammengesetzte Eigenschaft der Elemente } x \in \mathsf{M} \end{array} \tag{1}$$

Der Wert einer zusammengesetzten Eigenschaft ergibt sich aus dem Tupel $(f_1(x), ..., f_n(x))$, das die Basiswerte der Basiseigenschaften f_i für das Element x enthält.

Elementeigenschaft Die Eigenschaft g eines Elementes x wird mit $x.g$ bezeichnet. Die zusammengesetzte Eigenschaft f eines Elementes x wird mit $x.f$ oder $x.(f_1, ..., f_n)$ bezeichnet.

2.2 Algorithmen

Beziehungen zwischen Elementeigenschaften Der Wert einer Eigenschaft eines Elementes ergibt sich aus einer Eingabe des Benutzers oder wird mit einem Algorithmus aus den Werten anderer Eigenschaften des selben oder anderer Elemente berechnet. Dadurch ergeben sich Abhängigkeiten zwischen den Elementeigenschaften.

Algorithmen Die Werte von abhängigen Eigenschaften eines oder mehrerer Elemente werden mit einem Algorithmus bestimmt. Die Eingabeparameter eines Algorithmus ist ein m-Tupel von Elementeigenschaften. Die Ausgabe ist ein n-Tupel von Elementeigenschaften.

$$(f_1, \cdots, f_m) \quad \longrightarrow \quad \boxed{\mathsf{A}} \quad \longrightarrow \quad (h_1, \cdots, h_n) \tag{2}$$

Vollständige Bindungsrelation Die Abhängigkeiten zwischen den Elementeigenschaften werden durch Algorithmen bestimmt. Die Menge aller Elementeigenschaften einer Applikation sei F und die Menge aller Algorithmen einer Applikation sei A. Ist eine Eigenschaft f eines Elementes e Eingabeparameter eines Algorithmus A, so ist das Paar $(e.f, A)$ Element der vollständigen Bindungsrelation B. Ist eine Eigenschaft f eines Elementes e Ausgabeparameter eines Algorithmus A, so ist das Paar $(A, e.f)$ Element der vollständigen Bindungsrelation B. Die vollständige Bindungsrelation B ist ein bipartiter Graph.

$$\mathsf{B} := (\mathsf{A}, \mathsf{F}, R_e, R_a) \tag{3}$$

$$\begin{array}{ll} R_e \subseteq \mathsf{F} \times \mathsf{A} & \text{Eingabeparameter, Algorithmen} \\ R_a \subseteq \mathsf{A} \times \mathsf{F} & \text{Algorithmen, Ausgabeparameter} \end{array} \tag{4}$$

Direkte Abhängigkeit zwischen Elementeigenschaften Die Menge aller Elementeigenschaften sei F. Gibt es einen Algorithmus, der die Eigenschaft f eines Elementes b als Eingabeparameter und die Eigenschaft g des Elementes a als Ausgabeparameter besitzt, so heißt die Eigenschaft $b.f$ von der Eigenschaft $a.g$ direkt abhängig. Die geordneten Paare $(a.g, b.f)$ sind die Elemente der Bindungsrelation der Elementeigenschaften $\mathsf{B_F}$.

$$\begin{array}{lll} \mathsf{B_F} & \subseteq & \mathsf{F} \times \mathsf{F} \\ \mathsf{B_F} & & \text{Bindungsrelation der Elementeigenschaften} \end{array} \tag{5}$$

Eine Eigenschaft kann nicht von sich selbst abhängig sein. Die Bindungsrelation der Elementeigenschaften $\mathsf{B_F}$ ist deshalb antireflexiv.

$$\mathsf{B_F} := \{(a.g, b.f) \in \mathsf{F} \times \mathsf{F} \mid \begin{array}{l} \text{Eigenschaft } f \text{ des Elementes } b \text{ ist direkt abhängig} \\ \text{von der Eigenschaft } g \text{ des Elementes } a\} \end{array} \tag{6}$$

Die Relation $\mathsf{B_F}$ kann aus der vollständigen Bindungsrelation B abgeleitet werden.

$$\begin{array}{lll} \mathsf{B_F} & = & \{(a, b) \in \mathsf{F} \times \mathsf{F} \,|(a, b) \in \mathsf{B}^2\} \\ \mathsf{B_F} & \subset & \mathsf{B}^2 \end{array} \tag{7}$$

Indirekte Abhängigkeit von Elementeigenschaften Eine Eigenschaft f eines Elementes b ist indirekt von der Eigenschaft g eines Elementes a abhängig, wenn mindestens ein Weg von $a.g$ nach $b.f$ im Graphen von $\mathsf{B_F}$ existiert und $b.f$ von $a.g$ nicht direkt abhängig ist.

Direkte Beziehung zwischen Algorithmen Die Menge aller Algorithmen sei A. Ist eine Eigenschaft f Ausgabeparameter eines Algorithmus A und Eingabeparameter eines Algorithmus B, so stehen die Algorithmen A und B direkt in Beziehung. Die geordneten Paare (A, B) sind die Elemente der Relation $\mathsf{B_A}$.

$$\mathsf{B_A} := \{(A,B) \in \mathsf{A} \times \mathsf{A} \mid \begin{array}{l}\text{mindestens ein Ausgabeparameter des Algorithmus } A \\ \text{ist Eingabeparameter des Algorithmus } B\ \}\end{array} \tag{8}$$

Die Relation $\mathsf{B_A}$ kann aus der vollständigen Bindungsrelation B abgeleitet werden.

$$\begin{array}{rcl}\mathsf{B_A} & = & \{(a,b) \in \mathsf{A} \times \mathsf{A} \,|(a,b) \in \mathsf{B}^2\} \\ \mathsf{B_A} & \subset & \mathsf{B}^2\end{array} \tag{9}$$

System Gegeben sei eine Menge E von Elementen, eine Menge A von Algorithmen, eine Menge F von Eigenschaften der Elemente und die vollständige Bindungsrelation B. Das Gebilde $(\mathsf{E}, \mathsf{A}, \mathsf{F}; \mathsf{B})$ heißt ein System und wird mit S bezeichnet.

$$\mathsf{S} := (\mathsf{E}, \mathsf{A}, \mathsf{F}; \mathsf{B}) \tag{10}$$

2.3 Aktualisierung des Systems

Zielmenge Bei einer Aktualisierung (Update) des Systems werden die abhängigen Elementeigenschaften neu berechnet. Die Menge der zu aktualisierenden Elementeigenschaften umfaßt nicht unbedingt alle abhängigen Elementeigenschaften eines Systems, sondern wird durch Auswahl bestimmt. Die ausgewählten Elementeigenschaften sind die Elemente der Zielmenge Z des Updates :

$$\mathsf{Z} := \{a \in \mathsf{F} \mid a \text{ wird für das Update ausgewählt}\} \tag{11}$$

Domäne der Elementeigenschaften Die Elemente der Zielmenge sind u.U. von Elementen abhängig, die nicht in die Auswahl aufgenommen wurden. Da aber die Elemente in Z von diesen abhängen, müssen auch diese aktualisiert werden. Die Menge aller Elementeigenschaften, die bei der Aktualisierung neu zu berechnen sind, heißt Domäne der Elementeigenschaften und wird mit $\mathsf{D_F}$ bezeichnet :

$$\mathsf{D_F} := \{a \in \mathsf{F} \mid a \text{ muß aktualisiert werden}\} \tag{12}$$

Domäne der Algorithmen Der Wert jeder abhängigen Elementeigenschaft wird mit einem Algorithmus des Systems neu berechnet. Bei der Aktualisierung eines Systems müssen diejenigen Algorithmen ausgeführt werden, die die Werte der Elementeigenschaften der Domäne $\mathsf{D_F}$ neu berechnen. Die Menge der bei einer Aktualisierung auszuführenden Algorithmen heißt Domäne der Algorithmen und wird mit $\mathsf{D_A}$ bezeichnet.

$$\mathsf{D_A} := \{A \in \mathsf{A} \mid \begin{array}{l}\text{Algorithmus } A \text{ muß der Aktualisierung} \\ \text{der Elemente in } \mathsf{D_F} \text{ ausgeführt werden }\}\end{array} \tag{13}$$

Domäne der Aktualisierung Die Vereinigung der Mengen $\mathsf{D_A}$ und $\mathsf{D_F}$ heißt Domäne der Aktualisierung und wird mit D bezeichnet.

$$\mathsf{D} = \mathsf{D_A} \cup \mathsf{D_F} \tag{14}$$

Bestimmen der Domäne der Aktualisierung Ausgehend von den für das Update ausgewählten Eigenschaften wird im Graphen der Bindungsrelation B eine Breiten- oder Tiefensuche durchgeführt. Ist der besuchte Knoten des bipartiten Graphen von B eine Elementeigenschaft, so wird diese in die Domäne der Elementeigenschaften $\mathsf{D_F}$ aufgenommen. Ist der besuchte Knoten des Graphen von B ein Algorithmus, so wird dieser in die Domäne der Algorithmen $\mathsf{D_A}$ aufgenommen.

Reihenfolge der Aktualisierung Ist eine Elementeigenschaft a von der Eigenschaft b abhängig, so muß zuerst b aktualisiert werden. Da es von b nach a mehrere Wege im Graphen von B geben kann, ist die Reihenfolge der Aktualisierung nicht beliebig.

Reduzierte Bindungsrelation Die vollständige Bindungsrelation B wird auf die Paare reduziert, deren Komponenten Elemente der Domäne D sind. Dazu wird die Reduktionsmatrix Γ aufgestellt. Die Reduktionsmatrix Γ enthält Einsvektoren, die an der entsprechenden Stelle eines Elements aus D den Wert Eins (wahr) besitzen.

$$\begin{gathered} \bar{\mathsf{B}} = \Gamma^T \mathsf{B} \Gamma \\ \bar{\mathsf{B}} := \{(a,b) \in \mathsf{D} \times \mathsf{D} \mid (a,b) \in \mathsf{B}\} \end{gathered} \tag{15}$$

Reduzierte Bindungsrelation der Elementeigenschaften Die reduzierte Bindungsrelation der Elementeigenschaften $\bar{\mathsf{B}}_\mathsf{F}$ enthält nur Paare (a,b) mit der Eigenschaft, daß sowohl a als auch b Element der Domäne der Elementeigenschaften $\mathsf{D_F}$ ist.

$$\bar{\mathsf{B}}_\mathsf{F} := \{(a,b) \in \mathsf{D_F} \times \mathsf{D_F} \mid (a,b) \in \mathsf{B}^2\} \tag{16}$$

Die reduzierte Bindungsrelation der Elementeigenschaften $\bar{\mathsf{B}}_\mathsf{F}$ läßt sich aus der Bindungsrelation B bestimmen. Die Reduktionsmatrix Γ_F enthält Einsvektoren, die an der entsprechenden Stelle eines Elements aus $\mathsf{D_F}$ den Wert Eins (wahr) besitzen.

$$\bar{\mathsf{B}}_\mathsf{F} = \Gamma_F^T \mathsf{B}^2 \Gamma_F \tag{17}$$

Reduzierte Relation $\bar{\mathsf{B}}_\mathsf{A}$ Die reduzierte Relation $\bar{\mathsf{B}}_\mathsf{A}$ enthält nur Paare (a,b) mit der Eigenschaft, daß sowohl a als auch b Element der Domäne der Algorithmen $\mathsf{D_A}$ ist.

$$\bar{\mathsf{B}}_\mathsf{A} := \{(a,b) \in \mathsf{D_A} \times \mathsf{D_A} \mid (a,b) \in \mathsf{B}^2\} \tag{18}$$

Die reduzierte Relation $\bar{\mathsf{B}}_\mathsf{A}$ läßt sich aus der Bindungsrelation B bestimmen. Die Reduktionsmatrix Γ_A enthält Einsvektoren, die an der entsprechenden Stelle eines Elements aus $\mathsf{D_A}$ den Wert Eins besitzen.

$$\bar{\mathsf{B}}_\mathsf{A} = \Gamma_A^T \mathsf{B}^2 \Gamma_A \tag{19}$$

Durchführung der Aktualisierung Die reduzierte Relation $\bar{B}_A$ wird topologisch sortiert. Das Ergebnis ist eine Folge von Algorithmen. Die Folge wird sequentiell durchlaufen. Jeder Algorithmus wird genau einmal ausgeführt.

3 Objektorientierte Systeme

Die in den vorhergehenden Abschnitten gezeigte formale Beschreibung eines Systems wird im folgenden auf objektorientierte Softwaresysteme übertragen.

3.1 Elemente

Objektorientierte Softwaresysteme (Anwendungen) speicher Daten in strukturierten Mengen, deren Elemente Objekte sind. Die Elemente eines objektorientierten Softwaresystems sind Datenelemente, Datenmengen und Objekte.

Datenelemente Eine in einem Rechner persistent oder temporär gespeicherte Informationseinheit heißt Datenelement, wenn diese von allen anderen Informationseinheiten trennbar sowie eindeutig identifizierbar ist und nicht aus anderen Datenelementen zusammengesetzt ist. Dies trifft für die primitiven Datentypen einer Plattform und für Referenzen auf Objekte zu. Ein Datenelement speichert den Wert genau einer Eigenschaft des Datenelementes.

Objekte Jedes Objekt ist eindeutig identifizierbar. Ein Objekt besteht aus Methoden und Attributen. Die Attribute eines Objektes sind Datenelemente oder Datenmengen. Jedes Attribut speichert genau eine Eigenschaft eines Objektes. Die Attribute eines Objektes definieren eine zusammengesetzte Eigenschaft für das Objekt.

Datenmengen Die Zusammenfassung von Datenelementen zu einer Menge wird als Datenmenge bezeichnet. Eine Datenmenge wird durch Aufzählen der Identifikatoren ihrer Elemente definiert. Im Rechner wird eine Datenmenge entweder als Feld oder als Objekt gespeichert.

Identifizierung von Elementen Jedes Objekt wird mit einem systemweit eindeutigem Namen identifiziert. Jedes Attribut wird mit einem für das Objekt eindeutigem Namen identifiziert. Ein Attribut wird systemweit eindeutig identifiziert, indem der Identifikator des Objektes und der Identifikator des Attributes gekettet werden.

Referenzen Eine Referenz ist eine Eigenschaft eines Objektes. Der Zweck einer Referenz ist es, den Pfad zu Eigenschaften anderer Objekte festzulegen. Eine Referenz x eines Elementes e wird mit $e.x$ bezeichnet. Der Wert der Eigenschaft $e.x$ identifiziert ein Objekt. Besitzt das referenzierte Objekt eine Eigenschaft f, so kann diese Eigenschaft mit $e.x.f$ identifiziert werden. Ist die Eigenschaft $e.x.f$ wiederum eine Referenz, so läßt sich die Kette beliebig fortsetzen.

3.2 Algorithmen

Methoden Die Algorithmen des Systems sind in den Methoden der Objekte implementiert. Nach dem Prinzip der Datenkapselung (data hiding) ist es zweckmäßig, auch den lesenden und schreibenden Zugriff auf ein Objektattribut über eine Methode vorzunehmen.

Identifizierung von Methoden Jede Methode besitzt einen für das Objekt eindeutigen Namen. Eine Methode wird systemweit eindeutig identifiziert, indem der Identifikator des Objektes und der Name der Methode gekettet werden.

Eingabeparameter Diejenigen Eigenschaften, auf die eine Methode lesend zugreift, sind die Eingabeparameter der Methode. Die Eingabeparameter sind nicht auf die Übergabeparameter des Methodenaufrufes beschränkt.

Ausgabeparameter Diejenigen Eigenschaften, auf die die Methode schreibend zugreift, sind die Ausgabeparameter der Methode. Die Ausgabeparameter sind nicht auf den Rückgabewert des Methodenaufrufes beschränkt.

Methoden zur Aktualisierung Methoden, die die Werte von abhängigen Eigenschaften aktualisieren, besitzen als Eingabeparameter Eigenschaften, die Teil des Systems sind. Diese Methoden sind bei der Aktualisierung eines Systems auszuführen.

Methoden zum Setzen von Elementeigenschaften Methoden, die Werte von Objektattributen setzen, besitzen ausschließlich Eingabeparameter, die nicht Teil des Systems sind. Diese Parameter sind bspw. Eingaben des Benutzers oder Werte, die aus einer Datenbank oder einer Datei gelesen werden. Der Zeitpunkt der Ausführung dieser Methoden wird vom Benutzer bestimmt. Die in diesen Methoden implementierten Algorithmen definieren keine Abhängigkeiten zwischen Elementeigenschaften und sind deshalb bei der Aktualisierung eines Systems nicht zu berücksichtigen.

Methoden zum Lesen von Elementeigenschaften Methoden, die Werte von Objektattributen lesen, besitzen ausschließlich Ausgabeparameter, die nicht Teil des Systems sind. Die Methoden werden nur für den Zugriff auf Elementeigenschaften benutzt und sind deshalb bei Aktualisierung eines Systems nicht zu berücksichtigen.

3.3 Aktualisierung einer Objektbasis

Implizite Eigenschaften Der Wert einer impliziten Eigenschaft eines Objektes wird zu dem Zeitpunkt berechnet, zu dem der Wert der Eigenschaft abgefragt wird. Der Wert der Eigenschaft wird nicht im Objekt gespeichert. Ein solches Attribut ist bezüglich der Attribute, von denen es direkt abhängig ist, immer aktuell.

Verzögert berechnete Eigenschaft Ein Attribut, das verzögert aktualisiert wird, besitzt eine Methode zur Aktualisierung des Attributes. Der berechnete Wert wird im System (i.d.R. im Objekt selbst) gespeichert. Wird auf den Wert der Eigenschaft zugegriffen, so wird von der Lese-Methode der gespeicherte Wert zurückgegeben. Um den Wert zu aktualisieren, muß vorab die Aktualisierungsmethode ausgeführt werden.

Änderung von Elementen Ein Datenelement gilt als geändert, wenn sich sein Wert geändert hat.

Eine Datenmenge gilt als geändert, wenn sich der Wert von mindestens einem der enthaltenen Elemente geändert hat, mindestens ein Element hinzugefügt wurde oder mindestens ein Element entfernt wurde.

Ein Objekt gilt als geändert, wenn sich der Wert mindestens einer Eigenschaft oder die Definition einer Methode des Objektes geändert hat.

Komponentenabhängigkeit Das Speichern einer Referenz in einem Objekt definiert eine Abhängigkeit des Objektes von einer anderen Komponente des Systems. Durch die gespeicherte Referenz wird ein Pfad zu einer Elementeigenschaft des refernzierten Objektes festgelegt. Bei einer Eigenschaft eines referenzierten Objektes ist nicht nur die Änderung der Eigenschaft sondern auch die Änderung des Pfades zu berücksichtigen.

Durchführung der Aktualisierung Die Aktualisierung eines objektorientierten Systems wird in folgenden Schritten durchgeführt:

1. Auswahl der zu aktualisierenden Elemente

   Die Elemente der Zielmenge Z werden vom Benutzer durch Auswahl bestimmt. Objekte werden in der Zielmenge durch ihre Attribute ersetzt. Datenmengen werden durch ihre Elemente ersetzt.

2. Bestimmen der Domäne der Aktualisierung und der reduzierten Bindungsrelation

   Die Domäne der Aktualisierung wird durch eine Breitensuche in der vollständigen Bindungsrelation B bestimmt. Jeder besuchte Knoten wird in die Domäne D eingetragen.

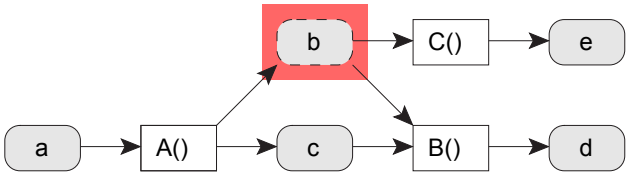
   Bei der Breitensuche in der vollständigen Bindungsrelation B wird die reduzierte Bindungsrelation $\bar{B}$ aufgebaut. Die Paare, die aus einem bei der Breitensuche besuchten Knoten und seinen Vorgängern gebildet werden können, werden als Paare in die reduzierte Bindungsrelation eingetragen.

3. Entfernen impliziter Attribute

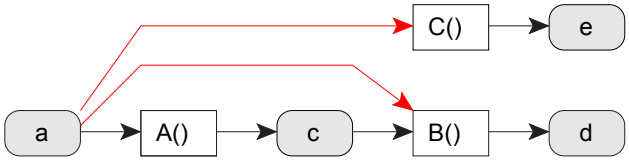
   Ein implizites Attribut wird beim Zugriff auf das Attribut unmittelbar berechnet. Sein Wert wird nicht im System gespeichert. Deshalb sind implizite Attribute aus der Domäne zu entfernen. Die Domäne der Aktualisierung ohne implizite Attribute wird mit D_v bezeichnet.

Alle Paare, die ein implizites Attribut enthalten, werden aus der reduzierten Bindungsrelation entfernt. Der Vorgänger eines impliziten Attributes ist die Methode M, die Nachfolger eines impliziten Attributes sind die Methoden M_i. Die Eingabeparameter der Methode M werden als Eingabeparameter der Methoden M_i eingetragen.

Beispiel :



$$\begin{aligned} \bar{\mathsf{B}} &= \{(a,A),(A,b),(A,c),(b,C),(b,B),(c,B),(C,e),(B,d)\} \\ \mathsf{D} &= \{a,A,b,c,C,B,e,d\} \end{aligned}$$

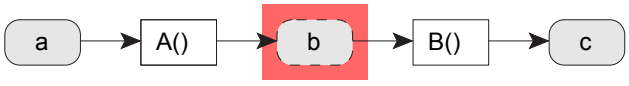


$$\begin{aligned} \bar{\mathsf{B}}_{\mathsf{v}} &= \bar{\mathsf{B}} \setminus \{(A,b),(b,C),(b,B)\} \cup \{(a,C),(a,B)\} \\ &= \{(a,A),(a,C),(a,B),(A,c),(c,B),(C,e),(B,d)\} \\ \mathsf{D}_{\mathsf{v}} &= \{a,A,c,C,B,e,d\} \end{aligned}$$

4. Entfernen von Methoden ohne Nachfolger

   Das Entfernen von impliziten Attributen hat zur Folge, daß im Graphen der Relation $\bar{\mathsf{B}}_{\mathsf{v}}$ Knoten existieren, die Methoden zugeordnet sind und keinen Nachfolger im Graphen von $\bar{\mathsf{B}}_{\mathsf{v}}$ besitzen. Diese Methoden besitzen nur implizite Attribute als Ausgabeparameter und müssen bei der Aktualisierung des Systems nicht ausgeführt werden. Diese Methoden werden aus der Domäne der Aktualisierung entfernt. Die Domäne der Aktualisierung ohne implizite Attribute und ohne Methoden ohne Nachfolger wird mit $\mathsf{D}_{\mathsf{v,m}}$ bezeichnet. Die Paare, die Methoden ohne Nachfolger im Graphen von $\bar{\mathsf{B}}_{\mathsf{v}}$ enthalten, werden aus der Relation $\bar{\mathsf{B}}_{\mathsf{v}}$ entfernt. Die sich daraus ergebebende reduzierte Bindungsrelation wird mit $\bar{\mathsf{B}}_{\mathsf{v,m}}$ bezeichnet.

   Beispiel :



$$\begin{aligned} \bar{\mathsf{B}} &= \{(a,A),(A,b),(b,B),(B,c)\} \\ \mathsf{D} &= \{a,A,b,B,c\} \end{aligned}$$

$$\begin{aligned} \bar{\mathsf{B}}_{\mathsf{v}} &= \{(a,A),(a,B),(B,c)\} \\ \mathsf{D}_{\mathsf{v}} &= \{a,A,B,c\} \end{aligned}$$

$$\begin{aligned} \bar{\mathsf{B}}_{\mathsf{v,m}} &= \{(a,B),(B,c)\} \\ \mathsf{D}_{\mathsf{v,m}} &= \{a,B,c\} \end{aligned}$$

5. Bestimmen der Reihenfolge der Aktualisierung

   Das Quadrat von $\bar{\mathsf{B}}_{\mathsf{v,m}}$ wird bestimmt. Aus dem Quadrat der reduzierten Bindungsrelation wird der Subgraph $\bar{\mathsf{B}}_{\mathsf{A}}$ bestimmt, der die Paare enthält, deren Komponenten Elemente der Domäne der Methoden sind.

   $$\bar{\mathsf{B}}_{\mathsf{A}} = \Gamma_A^T \bar{\mathsf{B}}_{\mathsf{v,m}}^2 \Gamma_A \tag{20}$$

   Die Relation $\bar{\mathsf{B}}_{\mathsf{A}}$ wird topologisch sortiert. Das Ergebnis ist eine Folge von Methoden. Die Folge wird mit a bezeichnet.

6. Ausführen der Aktualisierungsmethoden

   Die Methoden werden in der Reihenfolge ausgeführt, in der sie in a eingetragen sind. Eine Methode wird nur dann ausgeführt, wenn sich die Eingabeparameter seit dem letzten Ausführen der Methode geändert haben.

4 Implementierung

Die in den vorhergehenden Abschnitten dargestellte Beschreibung von allgemeinen objektorientierten Systemen wird im Rechner umgesetzt. Die Umsetzung soll folgende Voraussetzungen erfüllen :

- Die Implementierung erfolgt generalisiert für Objekte beliebigen Typs. Die Implementierung ist damit unabhängig von bestimmten Datenmodellen oder Anwendungsbereichen.
- Die Bindungen zwischen den Elementeigenschaften können statisch zur Entwicklungszeit einer Anwendung und dynamisch zur Laufzeit einer Anwendung festgelegt werden.
- Die Implementierung der entwickelten Schnittstellen ist für die Klassen einer Anwendung weitestgehend automatisch zu generieren. Der Entwicklungsaufwand und die Fehleranfälligkeit wird dadurch reduziert.

4.1 Objektmodell

Das Objektmodell beschreibt die zur Verfügung stehenden Datentypen und legt für die Objekte bestimmte Eigenschaften fest. Zu den Eigenschaften gehören die Identifizierung von Objekten, Attributen und Methoden, die Bereitstellung von Versionsnummern für Objekte und Attribute.

Identifizierung von Objekten, Attributen und Methoden Jedes Objekt wird durch eine im System eindeutige Zeichenkette identifiziert. Jede Methode und jedes Attribut wird mit einem für das Objekt eindeutige Zeichenkette identifiziert. Methoden und Attribute werden systemweit eindeutig durch die Kettung des Identifikators des Objektes und des Identifikators der Methode bzw. des Attributes identifiziert.

Attribute Ein Attribut eines Objektes speichert den Wert einer Eigenschaft des Objektes. Ein Objektattribut ist entweder ein primitives Attribut, eine Referenz, eine Menge oder eine Abbildung.

- Primitive Attribute (`primitive`)

  Primitive Attribute werden mit den primitiven Datentypen einer Programmiersprache gespeichert. Die vom Objektmodell zur Verfügung gestellten Datentypen werden auf eine Teilmenge der IDL (Interface Definition Language) Datentypen beschränkt. Es werden folgende Datentypen unterstützt:

`long`	für Ganzzahlen,	`double`	für Fließkommazahlen,
`string`	für Zeichenketten und	`boolean`	für Wahrheitswerte

Die genannten Datentypen werden bei der Implementierung auf entsprechende Datentypen der verwendeten Programmiersprache abgebildet. Ein primitives Attribut kann auch ein implizites Attribut sein. Sein Wert kann nicht direkt gesetzt werden, sondern bestimmt sich aus dem Rückgabewert einer Methode.

- Referenzen (`reference`)

  Eine Referenz ist ein Verweis auf ein anderes Objekt. Eine Referenz speichert den Identifikator des referenzierten Objektes. Der Datentyp einer Referenz ist `string`.

- Mengen (`set`)

  Eine Menge ist unstrukturiert. Die Reihenfolge der Aufzählung der Elemente ist ohne Bedeutung. Jedes Element ist höchstens einmal enthalten. Ein Element einer Menge kann vom Typ primitives Attribut, Referenz, Menge oder Abbildung sein.

- Abbildungen (`map`)

  Eine Abbildung ordnet jedem Element einer Menge A genau ein Element einer Menge B zu. Ein Element aus der Menge A heißt Schlüssel, ein Element aus der Menge B heißt Wert. Ein Schlüssel wird als Zeichenkette dargestellt. Die Interpretation des Schlüssels wird von der Implementierung übernommen. Ein Wert kann vom Typ primitives Attribut, Referenz, Menge oder Abbildung sein.

Versionsnummern Um die Änderung von Attributen effizient feststellen zu können, ist es zweckmäßig, für Attribute Versionsnummern vorzusehen. Die Versionsnummern sind ganzzahlige Werte vom Typ `long`. Wird der Wert eines Attributes geändert, so wird seine Versionsnummer um Eins erhöht. Um eine Änderung eines Elementes festzustellen, ist somit der Vergleich der Versionsnummern ausreichend.

4.2 Generalisierte Schnittstellen

Für das in den vorhergehenden Abschnitten beschriebene Konzept werden plattformunabhängige Schnittstellen für Systeme und ihre Elemente in der Beschreibungssprache IDL definiert. Die im folgenden aufgeführten Schnittstellen sind in einem Modul zusammengefasst. Bild 1 zeigt in einem Klassendiagramm eine Übersicht über die Schnittstellen.

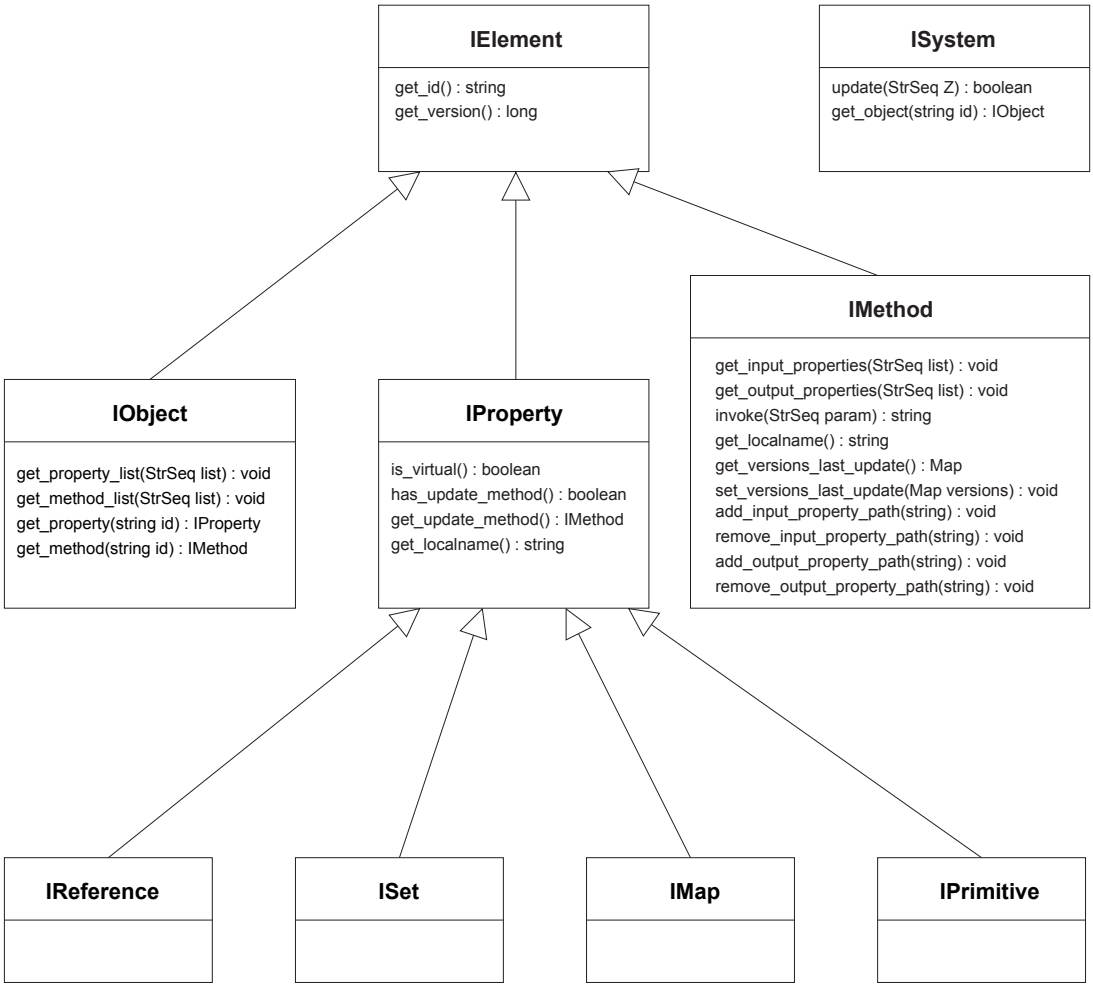


Abbildung 1: Generalisierte Schnittstellen

Erstellen von Abhängigkeiten zur Laufzeit Um Abhängigkeiten zwischen Attributen zur Laufzeit zu erstellen und zu entfernen, werden im Interface `IMethod` die Methoden `add_input_property_path(string)`, `add_output_property_path(string)`, `remove_input_property_path(string)` und `remove_output_property_path(string)` zur Verfügung gestellt.

4.3 Implementierung in C++

Die in Abschnitt 4.2 gezeigten Schnittstellen werden in der Programmiersprache C++ für Objekte beliebigen Typs implementiert. Das Objektmodell der Sprache C++ wird damit um das Hilfsmittel Reflexion erweitert.

In einem typischen Programmpaket aus dem Bereich des Ingenieurwesens sind eine Vielzahl von Klassen enthalten. Für jede ist die Schnittstelle zu implementieren. Um den Implementierungsaufwand zu reduzieren und die Fehleranfälligkeit zu verkleinern, wird ein Generator (`tmreflectc`) entwickelt, der die Implementierung der Schnittstellen automatisch erzeugt. Dazu wird das Schema einer Klasse im XML (eXtensible Markup Language) Format beschrieben.

Das Klassenschema steht im Argument des C++ Makros `_tm_ClassSchema_` im Quellcode der Klasse. Das Makro wird vom C++-Präprozessor durch die Deklaration der notwendigen Methoden ersetzt. Der Metacompiler `tmreflectc` extrahiert aus dem Quellcode einer Klasse das in XML beschriebenen Klassenschema und erzeugt aus dem Klassenschema die Implementierung der Schnittstellen in C++. Dieser Quellcode wird beim Übersetzen der Klasse dazugebunden.

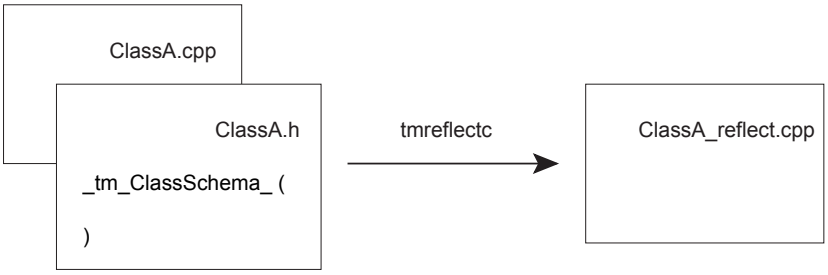


Abbildung 2: Metacompiler

5 Zusammenfassung

Objektorientierte Anwendungen aus dem Ingenieurwesen bestehen aus strukturierten Mengen, deren Elemente Objekte sind. Zwischen den Objekten bestehen vielfältige Abhängigkeiten. Die Beziehungen sind zur Zeit der Entwicklung einer Anwendung nur teilweise bekannt. Beziehungen zwischen Objekten müssen deshalb auch zur Laufzeit der Anwendung erzeugt und gelöscht werden können.

Aufgrund des hohen Rechenaufwandes wird die Objektbasis einer Anwendung verzögert aktualisiert. Der Benutzer wählt die zu aktualisierenden Elemente des Systems. Der Teil des Systems, der bei der Aktualisierung zu berücksichtigen ist, und die Reihenfolge der Aktualisierung kann mit Methoden der Graphentheorie bestimmt werden.

Die gezeigte Implementierung trägt der Dynamik der Beziehungen Rechnung. Die Umsetzung erfolgt generalisiert, d.h. für Objekte beliebigen Typs. Eine manuelle Implementierung der generalisierten Schnittstelle ist nicht zweckmäßig. Deshalb wird ein Generator entwickelt, der die Implementierung automatisiert vornimmt.